# 1 Robust linear regression

A. Consider the linear regression model:

$$y = w_0 + w_1 x. \tag{1}$$

For the given data set in this link, compute the unknown coefficients using the least squares solution (use the MLE/LS code developed in HW2). Predict the responses corresponding to the test inputs provided with the data[1].

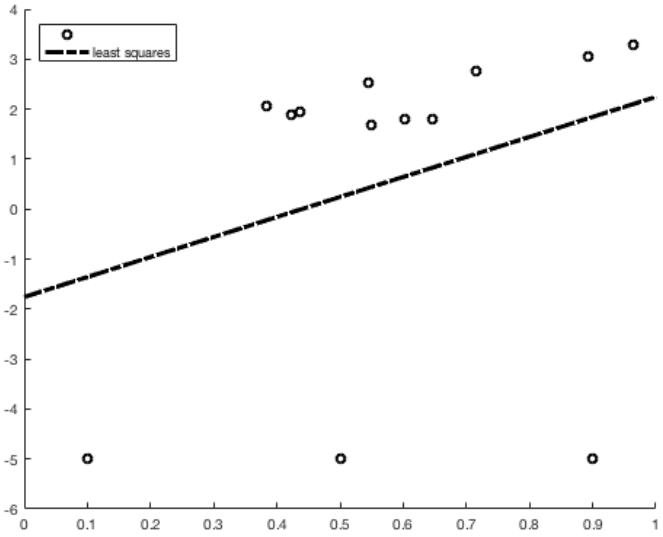
B. In part A of this problem, we observed that the least squares solution is not robust to outliers. One solution to this problem is to replace the Gaussian likelihood with a distribution that has heavy tails (e.g., Laplace, Student's t). Repeat the problem in part A with Laplace and Student's t distribution for the likelihood.

C. A more advanced loss-function is the Huber loss function. In this loss function, we combine both the $L_2$ and $L_1$ errors. Repeat the problem defined in part A using the Huber loss function (with $\delta = 1.0, 5.0$). Compare and discuss the results obtained in parts A, B and C.

A. using the least squares solution

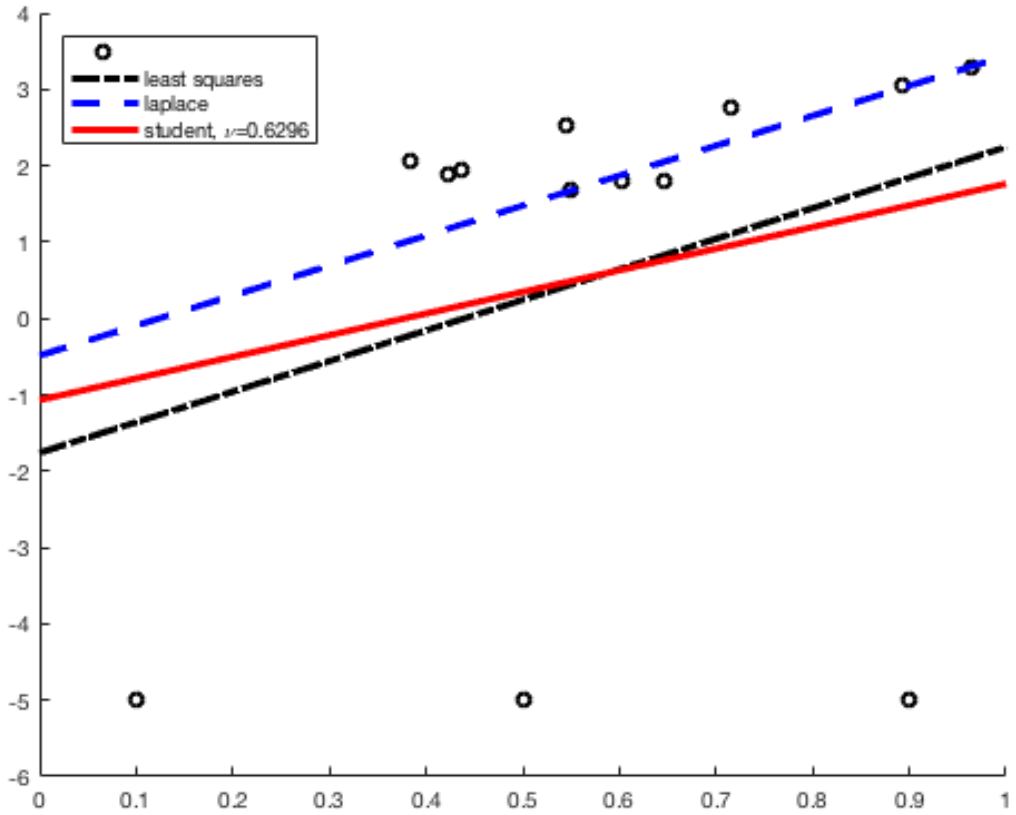
$$\begin{bmatrix} w_0 \\ w_1 \end{bmatrix} = W = W_{ML} = (X^T X)^{-1} X y$$

$$\begin{bmatrix} w_0 \\ w_1 \end{bmatrix} = W_{ML} = \begin{bmatrix} -1.7595 \\ 4.0041 \end{bmatrix}$$



B. Squared error penalizes deviations quadratically, so points far from line have more affect on the fit than points near the line.

To achieve robustness to outliers, one can replace the Gaussian with a distribution that has heavy tails (Laplace, Student's t). Such a distribution assigns higher likelihood to outliers, without having to perturb the regression line to "explain" them.



for Laplace

$$\begin{bmatrix} w_0 \\ w_1 \end{bmatrix} = \begin{bmatrix} -0.4859 \\ 3.9259 \end{bmatrix}$$

$$sigma2 = 429.7746$$

for Student's T

$$\begin{bmatrix} w_0 \\ w_1 \end{bmatrix} = \begin{bmatrix} -1.0691 \\ 2.8321 \end{bmatrix}$$

$$dof = 0.6296$$

$$sigma2 = 0.0688$$

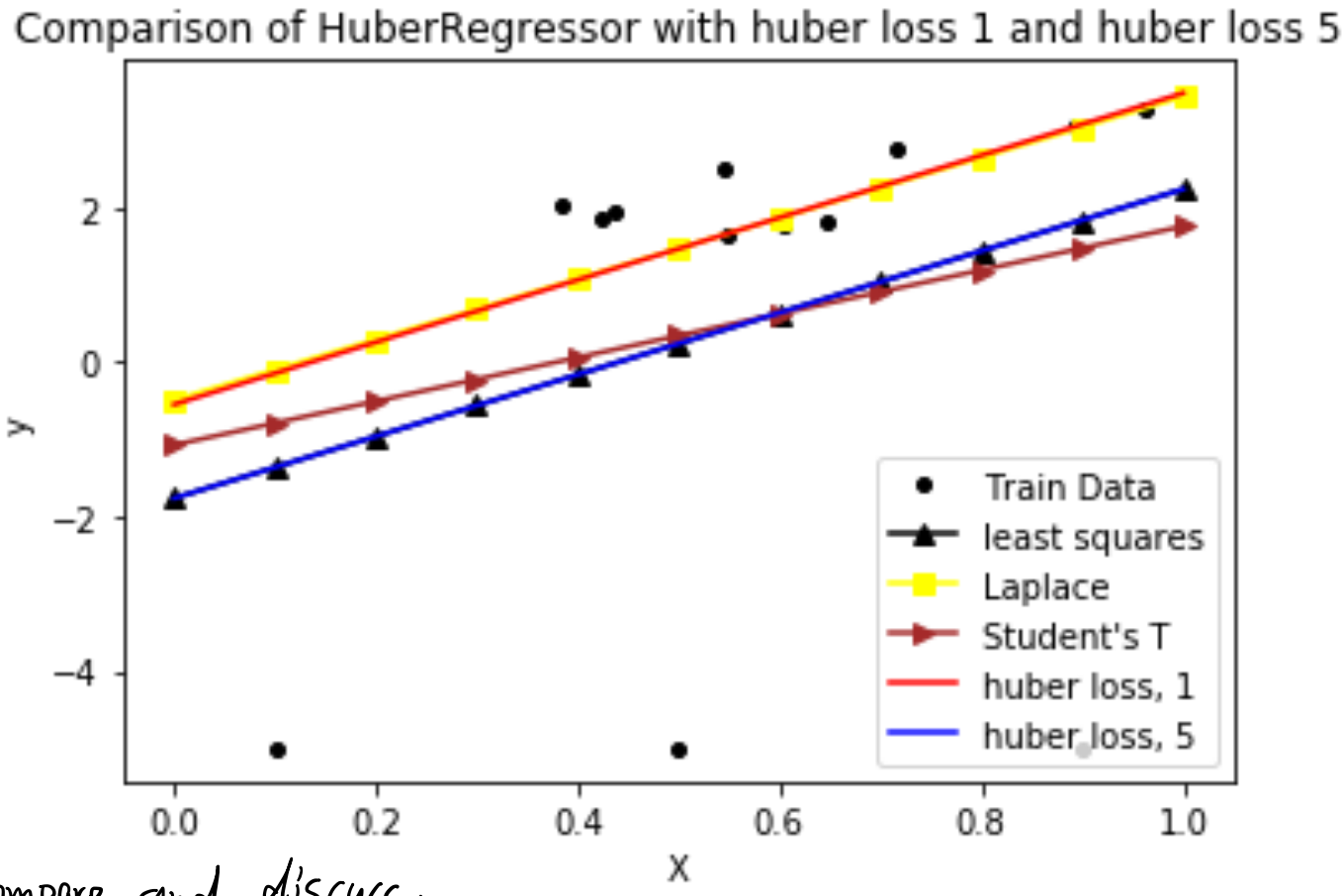
C.

optimizing the Huber loss is much faster than using the laplace likelihood.

We use HuberRegressor from sklearn.linear_model

$\delta = 1$ $W = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix} = \begin{bmatrix} -0.5448 \\ 4.0332 \end{bmatrix}$

$\delta = 5$, $W = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix} = \begin{bmatrix} -1.7595 \\ 4.0041 \end{bmatrix}$




Compare and discuss:

① huber loss, $\delta = 1$, the result is almost overlapped with Laplace

② huber loss, $\delta = 5$, the result is almost overlapped with least squares.

③ Laplace, and huber loss, $\delta = 1$ are the best to reduce the affect of outliers.

## 2 Online training in linear regression

A. For cases where we are dealing with a large data set, training conventional MLE becomes computationally expensive. Under such scenario, one option is to use the least mean squares (LMS) algorithm. As discussed in the class, LMS is an online algorithm where we process one data point at a time.

Consider a linear regression model as shown in Eq. 1. For the data-set given in this link, use LMS to learn the parameters $w_0$ and $w_1$. For learning the parameters, use one data at a time (i.e., online learning). Show the convergence plot. Run the algorithm up to 60 epochs. Discuss the results obtained,

B. Repeat the problem in part A by setting batch size = 5. Compare the convergence plots and discuss

A. If the error function comprises a sum over data points

$$E = \sum_n E_n$$

$$E_D(w) = \frac{1}{2}\sum_{n=1}^{N} (t_n - w^T \phi(x_n))^2$$

then after presentation of pattern, the stochastic gradient descent algorithm updates the parameter vector w using

$$w^{(\tau+1)} = w^{(\tau)} - \eta \nabla E_n$$

$$= w^{(\tau)} + \eta (t_n - w^{(\tau)T} \phi(x_n)) \phi(x_n)$$

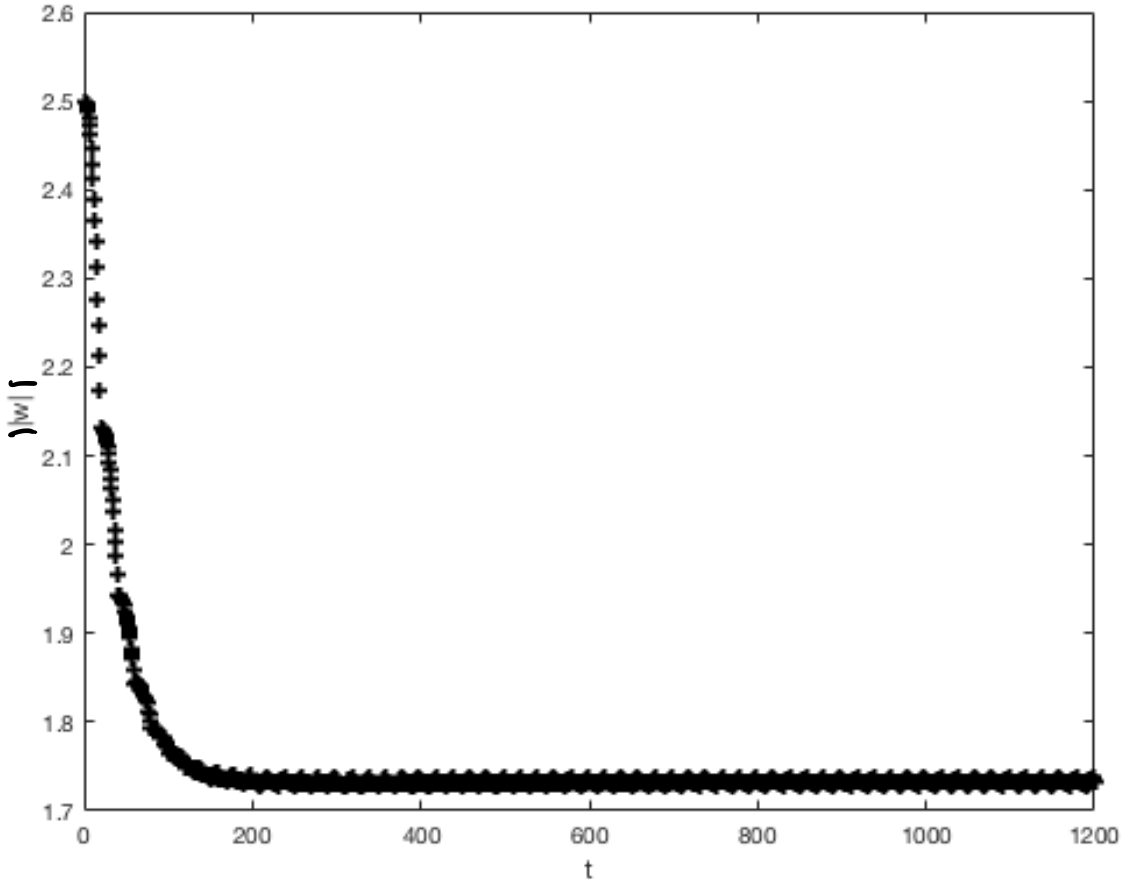
We first set $w^0 = \begin{bmatrix} 1.5 \\ 2 \end{bmatrix}$, and set $\eta = 0.0002$

then update the data one by one

$$w^{(\tau+1)} = w^{(\tau)} + \eta (y_n - w^{(\tau)T} \phi(x_n)) \phi(x_n)$$

in order to show the convergence,

we plot $||W||$ vs learning time $t$




After 60 epoches (every epoch updates 20 times)

$W_{b=1} = \begin{bmatrix} 1.4236 \\ 0.9871 \end{bmatrix}$ compared with the conventional

MLE method, $W_{LS} = \begin{bmatrix} 1.4097 \\ 0.9911 \end{bmatrix}$, $W$ is very close to $W_{LS}$

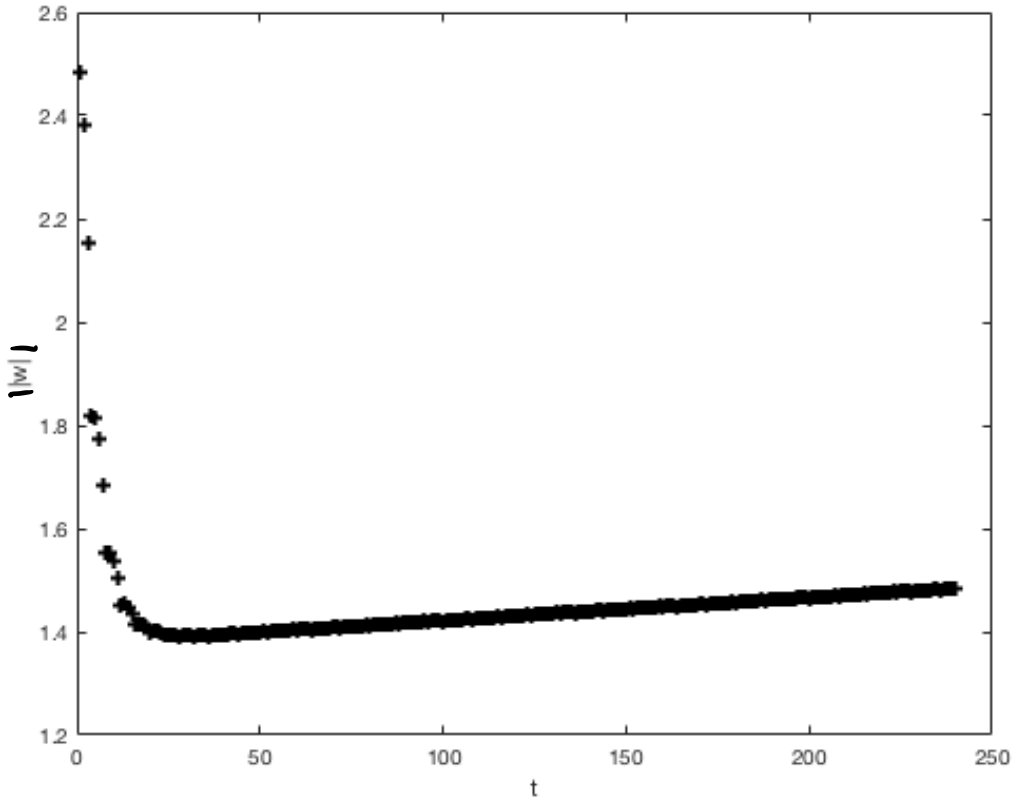
$|W|$ first converges very fast, then becomes flat.

And the result is very sensitive to learning rate $\eta$.

B.

Set batch size 5.

$$W^{\tau+1} = W^{\tau} + \eta \sum_{i=1}^{5} \left( y_{n+i} - W^{T}\phi(x_{n+i}) \right) \phi(x_{n+i})$$




After 60 epoches (every epoch updates 4 times)

$$W_{b=5} = \begin{bmatrix} 1.4720 \\ 0.1762 \end{bmatrix}$$

Therefore, setting batch size = 5, reduces the update times, but less accurate.

## 3 Least Squares Formalism to Classification

A. Consider a general classification problem with $K$ classes, with a $1-$of$-K$ binary coding scheme for the target vector $t$. Each class $C_k$ is described by its own linear model so that

$$y_k(\boldsymbol{x}) = \boldsymbol{w}_k^T \boldsymbol{x} + w_{k0}, \tag{2}$$

where $k = 1, \ldots, K$. We can conveniently group these together using vector notation so that

$$\boldsymbol{y}(\boldsymbol{x}) = \tilde{\mathbf{W}}^T \tilde{\boldsymbol{x}}, \tag{3}$$

where $\tilde{\mathbf{W}}$ is a matrix whose $k$-th column comprises the $D+1$-dimensional vector $\tilde{\boldsymbol{w}}_k = \left(w_{k0}, \boldsymbol{w}_k^T\right)^T$ and $\tilde{\boldsymbol{x}}$ is the corresponding augmented input vector $\left(1, \boldsymbol{x}^T\right)^T$ with a dummy input $x_0 = 1$. A new input $\boldsymbol{x}$ is then assigned to the class for which the output $y_k$ is the largest.

For the problem described above, evaluate the parameter matrix $\tilde{\boldsymbol{W}}$ by minimizing a sum-of-squares error function.

B. Write a code for least squares based classification (problem definition as described in part A). Use the data set given in this link. Find and plot the decision boundary. Provide your observations.

A. Consider a training data set $\{x_n, t_n\}$ where $n=1,\cdots,N$, and define a matrix $T$ whose $n^{th}$ row is the vector $t_n^T$, together with a matrix $\tilde{X}$ whose $n^{th}$ row is the $\tilde{x}_n^T$.

Therefore, the sum-of-squares error function can then be written as

$$E_D(\tilde{W}) = \frac{1}{2} Tr\left\{(\tilde{X}\tilde{W} - T)^T(\tilde{X}\tilde{W} - T)\right\}$$

Setting the derivative with respect to $\tilde{W}$ to zero

$$\frac{d E_D(\tilde{W})}{d\tilde{W}} = 0$$

This form is very similar to the linear Regression, least squares

$$E_D = \frac{1}{2}\sum_{n=1}^{N}\left(t_n - W^T\phi(x_n)\right)^2, \quad W = (\Phi^T\Phi)^{-1}\Phi t$$

therefore,

$$\tilde{W} = (\tilde{X}^T\tilde{X})^{-1}\tilde{X}^T T$$

(b),

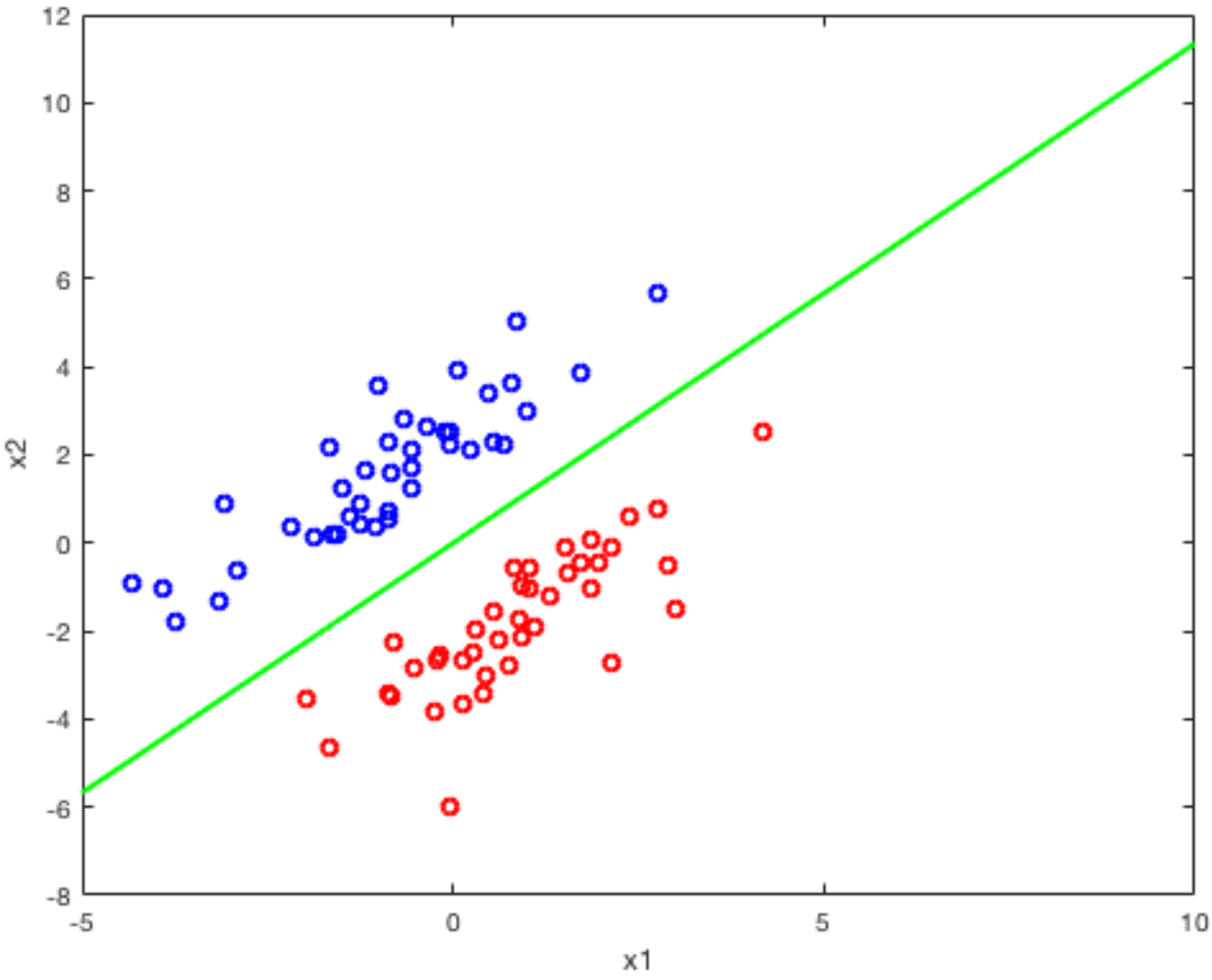
$$\tilde{X} = \begin{bmatrix} 1 & x_{11} & x_{12} \\ 1 & x_{21} & x_{22} \\ 1 & x_{31} & x_{32} \\ \vdots & \vdots & \vdots \\ 1 & x_{n1} & x_{n2} \end{bmatrix}$$

$$T = \begin{bmatrix} t_1 \\ t_2 \\ \vdots \\ t_n \end{bmatrix}$$

$$\tilde{W} = (\tilde{X}^T\tilde{X})^{-1}\tilde{X}^T T \xrightarrow{\text{Matlab}} \begin{bmatrix} 0.4884 \\ 0.1878 \\ -0.1655 \end{bmatrix} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix}$$

$w_0 + w_1 x_1 + w_2 x_2 = 0$

decision boundary



## 4 Logistic Regression for Multiclass Classification

A. Show that for a linearly separable data set, the maximum likelihood solution for the logistic regression model is obtained by finding a vector $\boldsymbol{w}$ whose decision boundary $\boldsymbol{w}^T\phi(\boldsymbol{x}) = 0$ separates the classes and then taking the magnitude of $\boldsymbol{w}$ to infinity.

B. The Hessian matrix for the logistic regression is given as:

$$\mathbf{H} = \left[\mathbf{\Phi}^T\mathbf{R}\mathbf{\Phi}\right],$$

where $\mathbf{\Phi}$ is the feature matrix, $\mathbf{R}$ is a diagonal matrix with elements $y_n(1-y_n)$ and $y_n$ is the output of the logistic regression model for input vector $x_n$.

Show that the Hessian matrix $\mathbf{H}$ is positive definite. Hence show that the error function is a convex function of $\boldsymbol{w}$ and that it has a unique minimum.

C. Write a code for implementing the iterative rewrighted least squares algorithm for logistic regression. Use this to find and plot the decision boundary corresponding to the data set given in this link. Compare the results with those obtained using least squares based classification.

A. For a data set $\{\phi_n, t_n\}$, where $t_n \in \{0,1\}$ and $\phi_n = \phi(x_n)$ with $n=1,\cdots,N$, the likelihood function can be written

$$p(t|w) = \prod_{n=1}^{N} y_n^{t_n}\{1-y_n\}^{1-t_n}$$

$t = (t_1,\cdots,t_N)^T$, $y_n = p(C_1|\phi_n)$

$w^T\phi(x)=0$ separates the classes, $C_1$, $C_2$

① when $\phi_n \in C_1$, $t_n = 1$, $w^T\phi_n > 0$

$$y_n^{t_n}(1-y_n)^{1-t_n} = y_n^{t_n} = P(C_1|\phi_n) = \frac{1}{1+e^{-w^T\phi_n}} \longrightarrow 1$$

$$w \to \infty, \quad \frac{1}{1+e^{-w^T\phi_n}} \to 1$$

② when $\phi_n \in C_2$, $t_n = 0$, $w^T\phi_n < 0$

$$y_n^{t_n}(1-y_n)^{1-t_n} = 1-y_n = 1 - \frac{1}{1+e^{-w^T\phi_n}} \longrightarrow 1$$

$$w \to \infty, \quad \frac{1}{1+e^{-w^T\phi_n}} \to 0$$

$$1 - \frac{1}{1+e^{-w^T\phi}} \to 1$$

Therefore, $w \to \infty$

B.

$$y_n = \sigma(w^T \phi_n)$$

$$0 < y_n < 1$$

$$y_n(1-y_n) > 0$$

for any non-zero vector $u \neq 0$

$$u^T H u = u^T \left[ \sum_{n=1}^{N} y_n (1-y_n) \phi_n \phi_n^T \right] u$$

$$= \sum_{n=1}^{N} y_n (1-y_n) u^T \phi_n [u^T \phi_n]^T$$

Let $b_n = (u^T \phi_n)^T$

$$u^T H u = \sum_{n=1}^{N} y_n (1-y_n) b_n^T b_n$$

$$= \sum_{n=1}^{N} y_n (1-y_n) b_n^2$$

for $n = 1, \cdots, N$, $\phi_n$ should belong to 2 classes.

Therefore, even if $\forall \phi_i \in C_1$, and $u^T \phi_i = 0$, there must exist $\phi_j \in C_2$, and $\phi_j \neq k\phi_i$, $u^T \phi_j \neq 0$

Therefore, there should be at lease one $b_n^2 \neq 0$.

then

$$u^T H u > 0$$

H is positive definite.

Since $H = \nabla\nabla E(w) > 0$

It follows that the error function is a concave function of W and that it has a minimum.

(C)

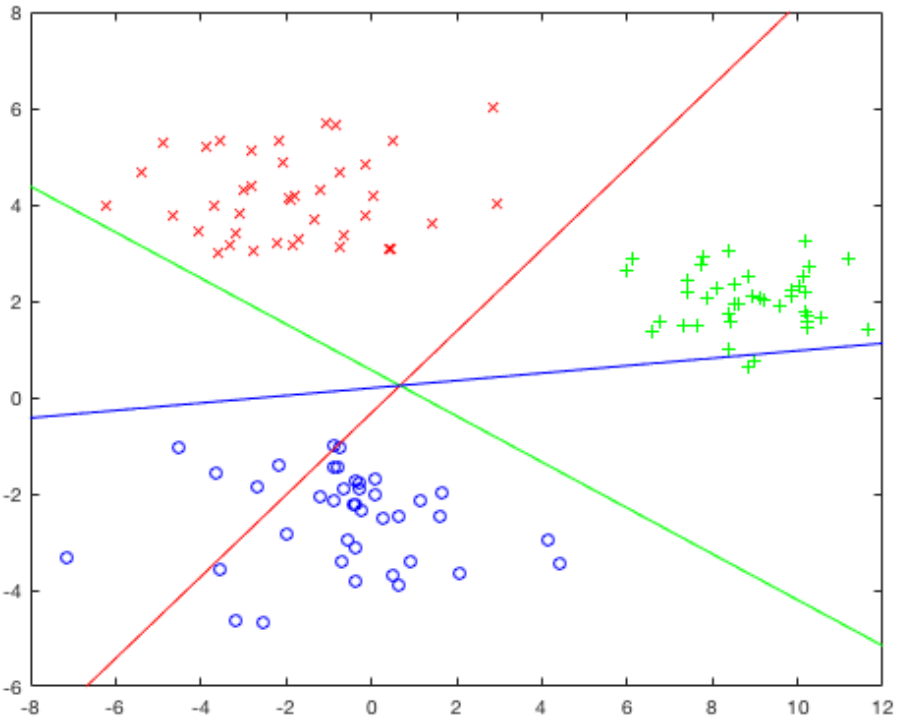
$$W = \begin{bmatrix} 0.8385 & -0.1841 & 4.8838 \\ -2.2737 & 2.7321 & 1.4484 \\ 2.4352 & 0.4521 & -3.3323 \end{bmatrix}$$

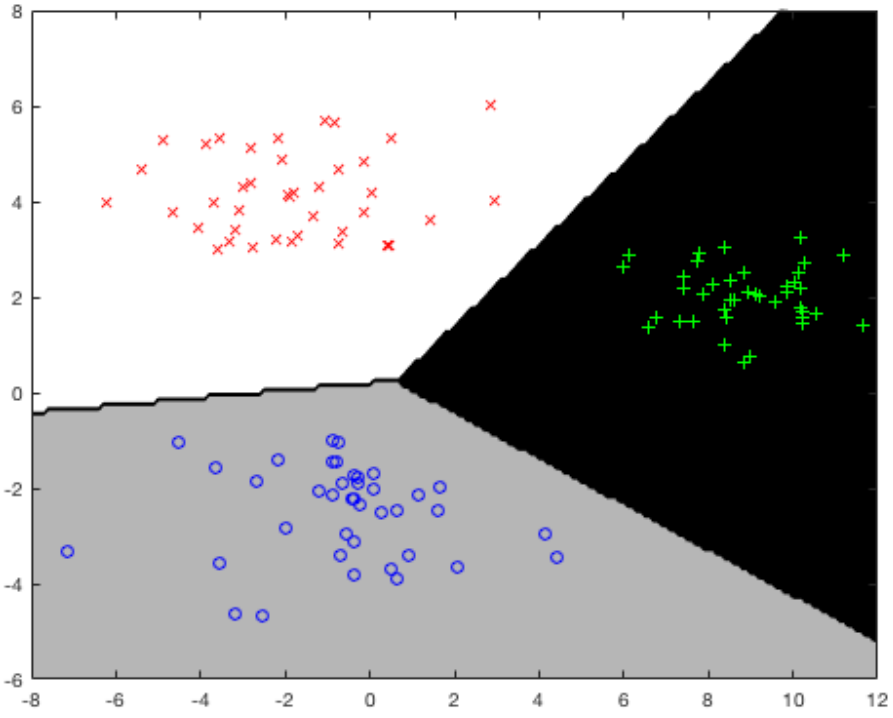
$$y_1 = -\frac{w(1,2) - w(2,2)}{w(1,3) - w(2,3)} x - \frac{w(1,1) - w(2,1)}{w(1,3) - w(2,3)}$$

$$y_2 = -\frac{w(2,2) - w(3,2)}{w(2,3) - w(3,3)} x - \frac{w(2,1) - w(3,1)}{w(2,3) - w(3,3)}$$

$$y_3 = -\frac{w(1,2) - w(3,2)}{w(1,3) - w(3,3)} x - \frac{w(1,1) - w(3,1)}{w(1,3) - w(3,3)}$$

using the bishop matlab code.


8
6
4
2
0
-2
-4
-6
-8
-6
-4
-2
0
2
4
6
8
10
12


8
6
4
2
0
-2
-4
-6
-8
-6
-4
-2
0
2
4
6
8
10
12

## 5 Fishers Discriminant

A. Consider a two-class problem in which there are $N_1$ points of class $\mathcal{C}_1$ and $N_2$ points of class $\mathcal{C}_2$. Therefore, the mean vector of the two classes are given by

$$\boldsymbol{m}_1 = \frac{1}{N_1}\sum_{n\in\mathcal{C}_1}\boldsymbol{x}_n, \qquad \boldsymbol{m}_2 = \frac{1}{N_2}\sum_{n\in\mathcal{C}_2}\boldsymbol{x}_n. \tag{4}$$

The simplest measure of the separation of the classes, when projected onto $\boldsymbol{w}$ , is the separation of the projected class means. This suggest we might choose $\boldsymbol{w}$ so as to maximize

$$m_2 - m_1 = \boldsymbol{w}^T(\boldsymbol{m}_2 - \boldsymbol{m}_1) \tag{5}$$

where

$$m_k = \boldsymbol{w}^T\boldsymbol{m}_k$$

Show that maximization of the class separation criterion defined in Eq. 5 with respect to w, using a Lagrange multiplier to enforce the constraint $\boldsymbol{w}^T\boldsymbol{w} = 1$, leads to the result that $\boldsymbol{w} \propto (\boldsymbol{m}_2 - \boldsymbol{m}_1)$

B. The Fisher criterion is defined to be the ratio of the between-class variance to the within-class variance and is given by

$$J(\boldsymbol{w}) = \frac{(m_2 - m_1)^2}{s_1^2 + s_2^2} \tag{6}$$

where

$$y = \boldsymbol{w}^T\boldsymbol{x}$$

$$\boldsymbol{m}_1 = \frac{1}{N_1}\sum_{n\in\mathcal{C}_1}\boldsymbol{x}_n, \qquad \boldsymbol{m}_2 = \frac{1}{N_2}\sum_{n\in\mathcal{C}_2}\boldsymbol{x}_n.$$

$$m_k = \boldsymbol{w}^T\boldsymbol{m}_k$$

and

$$s_k^2 = \sum_{n\in\mathcal{C}_k}(y_n - m_k)^2$$



. Show that Eq. 6 can be written as:

$$\boldsymbol{J}(\boldsymbol{w}) = \frac{\boldsymbol{w}^T\mathbf{S}_B\boldsymbol{w}}{\boldsymbol{w}^T\mathbf{S}_W\boldsymbol{w}}, \tag{7}$$

where

$$\mathbf{S}_B = (\boldsymbol{m}_2 - \boldsymbol{m}_1)(\boldsymbol{m}_2 - \boldsymbol{m}_1)^T$$

and

$$\mathbf{S}_W = \sum_{n\in\mathcal{C}_1}(\boldsymbol{x}_n - \boldsymbol{m}_1)(\boldsymbol{x}_n - \boldsymbol{m}_1)^T + \sum_{n\in\mathcal{C}_2}(\boldsymbol{x}_n - \boldsymbol{m}_2)(\boldsymbol{x}_n - \boldsymbol{m}_2)^T$$

C. Write a code for Fisher's Linear Discriminant Analysis (FLDA). For the data set given in this link,

(a) Compute the Fisher's vector.

(b) Using FLDA, project the data to 2d. Show the plot and discuss (e.g., how many classes do you observe).

A.

maximize $f(w) = w^T(\vec{m}_2 - \vec{m}_1)$

subject to $g(w) = (w^T w - 1) = 0$ , since $w^T w = 1$

$$L(w, \lambda) = w^T(\vec{m}_2 - \vec{m}_1) + \lambda(w^T w - 1)$$

$$\frac{\partial L(w,\lambda)}{\partial \lambda} = w^T w - 1 = 0$$

$$\frac{\partial L(w,\lambda)}{\partial w} = (\vec{m}_2 - \vec{m}_1) + 2\lambda w = 0$$

$$w = -\frac{1}{2\lambda}(\vec{m}_2 - \vec{m}_1) \propto (\vec{m}_2 - \vec{m}_1)$$

B.

$$S_B = (\vec{m}_2 - \vec{m}_1)(\vec{m}_2 - \vec{m}_1)^T$$

$$S_W = \sum_{n \in C_1} (\vec{x}_n - \vec{m}_1)(\vec{x}_n - \vec{m}_1)^T + \sum_{n \in C_2} (\vec{x}_n - \vec{m}_2)(\vec{x}_n - \vec{m}_2)^T$$

$$w^T S_B w = w^T(\vec{m}_2 - \vec{m}_1)(\vec{m}_2 - \vec{m}_1)^T w$$

$$= [w^T(\vec{m}_2 - \vec{m}_1)][w^T(\vec{m}_2 - \vec{m}_1)]^T$$

$$W^T S_B W = \| W^T (\vec{m_2} - \vec{m_1}) \|^2$$

$$W^T S_B W = \| m_2 - m_1 \|^2$$

$$W^T S_B W = (m_2 - m_1)^2$$

$$W^T S_W W = \sum_{n \in C_1} W^T (\vec{x_n} - \vec{m_1}) (\vec{x_n} - \vec{m_1})^T W^T + \sum_{n \in C_2} W^T (\vec{x_n} - \vec{m_2}) (x_n - m_2)^T W^T$$

$$= \sum_{n \in C_1} \| W^T (\vec{x_n} - \vec{m_1}) \| + \sum_{n \in C_2} \| W^T (\vec{x_n} - m_2) \|$$

since $\vec{y} = W^T \vec{x}$

$$y_n = W^T \vec{x_n}$$

$$W^T S_W W = \sum_{n \in C_1} (y_n - m_1)^2 + \sum_{n \in C_2} (y_n - m_2)^2$$

$$= s_1^2 + s_2^2$$

Therefore, $J(W) = \dfrac{(m_2 - m_1)^2}{s_1^2 + s_2^2} = \dfrac{W^T S_B W}{W^T S_W W}$

C.

(a)

$$J(W) = \frac{|W\Sigma_B W^T|}{|W\Sigma_W W^T|}$$

$$\Sigma_B \triangleq \sum_c \frac{N_c}{N} (\mu_c - \mu)(\mu_c - \mu)^T$$

$$\Sigma_W \triangleq \sum_c \frac{N_c}{N} \Sigma_c$$

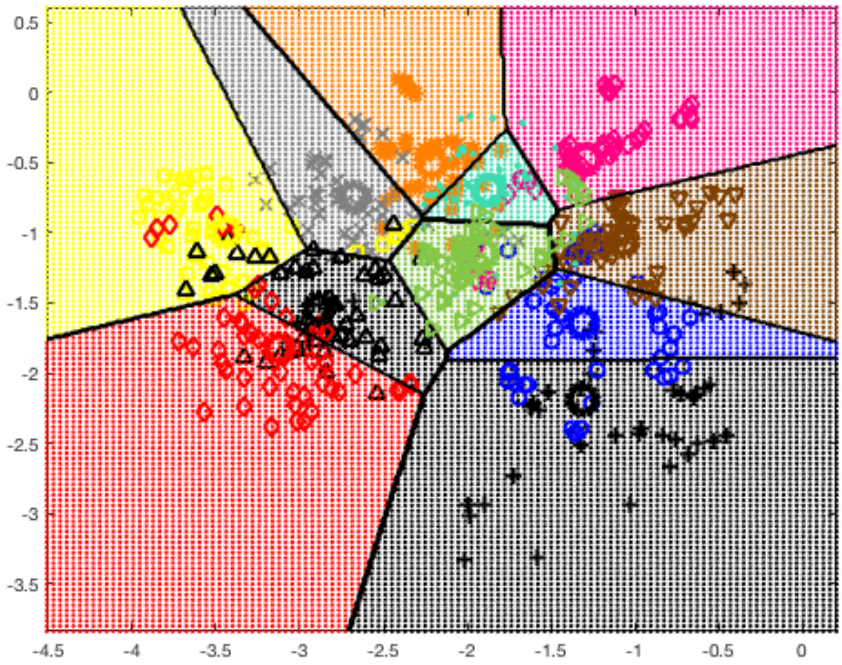
$$\Sigma_c = \triangleq \sum_{i:y_i=c} (x_i - \mu_c)(x_i - \mu_c)^T$$

$$W = \Sigma_W^{-\frac{1}{2}} W = [\quad]_{10\times 10}$$

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | -0.3573 | 0.3674 | 0.0742 | 0.1938 | 0.0065 | -0.3546 | -0.4340 | -0.2751 | 0.0974 | 0.0095 |
| 2 | 0.4545 | 0.1188 | -0.0423 | 0.3528 | -0.0326 | -0.3447 | -0.2951 | -0.4269 | 0.2080 | -0.1328 |
| 3 | 0.2130 | -0.1630 | 0.4562 | -0.0782 | -0.0059 | -0.2800 | -0.4820 | 0.0281 | 0.2600 | -0.3378 |
| 4 | 0.0097 | -0.2090 | 0.1156 | 0.3842 | 0.4940 | -0.0763 | -0.2100 | -0.5896 | 0.1167 | 0.1442 |
| 5 | -0.0031 | -0.5487 | -0.4292 | 0.4172 | 0.0744 | -0.2631 | -0.5387 | -0.0612 | -0.1701 | 0.0325 |
| 6 | 0.2798 | -0.5058 | 0.3041 | -0.0266 | -0.3512 | -0.5363 | -0.0691 | -0.2727 | 0.3684 | 0.0851 |
| 7 | 0.3333 | -0.2961 | 0.5067 | 0.4751 | 0.5612 | -0.2876 | -0.1737 | -0.0493 | -0.6672 | 0.2362 |
| 8 | 0.5158 | -0.3206 | -0.3457 | -0.4537 | 0.3528 | 0.0326 | -0.2286 | -0.1278 | -0.5014 | 0.1949 |
| 9 | 0.3813 | -0.1751 | -0.2403 | 0.2616 | -0.0193 | -0.1475 | -0.1812 | 0.5240 | 0.0186 | 0.8496 |
| 10 | 0.1394 | -0.0518 | -0.2460 | -0.0973 | 0.4314 | -0.4570 | 0.1945 | 0.1486 | -0.0666 | -0.1532 |
| 11 | | | | | | | | | | |

project the data to 2D

project on this 2 vector.




Observe 11 classes.

Using the pmtk3 matlab code

## 6 Bayesian logistic regression

Logistic regression corresponds to the following binary classification model

$$p(y|\boldsymbol{x}, \boldsymbol{w}) = \text{Ber}\left(y|\text{sigm}\left(\boldsymbol{w}^T\boldsymbol{x}\right)\right). \tag{8}$$

Consider a prior of the form

$$p(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{w}|\boldsymbol{0}, \mathbf{V}_0). \tag{9}$$

With this setup, we are interested to compute the full posterior $p(\boldsymbol{w}|\mathcal{D})$. However, unlike linear regression, this cannot be done analytically as there is no conjugate prior for this case. Hence, we have no option but to use approximations.

A. Provide a Gaussian approximation for the posterior.

B. Provide an approximation for the posterior predictive using a Monte Carlo approximation as well as the probit approximation.

C. For the data set given in this link, write a code for computing the posterior based on the Laplace approximation. Plot the log-likelihood, log-unnormalized posterior and Laplace approximation to the posterior.

D. Write a code for computing the posterior predictive distribution using MC approximation. Draw samples from the posterior predictive distribution and compute average over the samples. Recompute the posterior using probit approximation and compare the results.

A. $p(w) = N(w|0, V_0)$

$p(y|x,w) = \text{Ber}(y|\text{sigm}(w^T x))$

$p(w|D) \propto p(w)p(y|x,w)$

$$\log p(w|D) = -\frac{1}{2} w V_0^{-1} w + \sum_{n=1}^{N} \{ y_n \ln(\text{sigm}(w^T\phi_n)) + (1-y_n) \cdot \ln(1 - \text{sigm}(w^T\phi_n)) \} + \text{const}$$

To obtain a Gaussian approximation to the posterior distribution, we first maximize $\log p(w|D)$ to get the MAP (maximum posterior) solution $w_{MAP}$

$$w_{MAP} = \arg\max_w \log p(w|D)$$

$$V_N = -\nabla\nabla \log p(w|D)$$

$$= V_0^{-1} + \sum_{n=1}^{N} \text{sigm}(w^T\phi_n)(1 - \text{sigm}(w^T\phi_n))\phi_n\phi_n^T$$

the Gaussian approximation to the posterior distribution takes the form

$$p(w|D) \simeq N(w|w_{MAP}, V_N)$$

B. Monte Carlo approximation

$w^s \sim p(w|D)$ are samples for the posterior

$$p(y=1|x,D) \approx \frac{1}{S}\sum_{s=1}^{S} \text{sigm}((w^s)^T x)$$

S is the sample times

Since we made Gaussian approximation to the posterior $p(w|D)$, $p(w|D)$ can be treated as Gausslan distribution. We can directly draw independent samples from that Gaussian distribution. using standard methods. Sample S times and average them.

---

Probit approximation

$$p(y=1|x,D) \approx \int \text{sigm}(w^T x)\, p(w|D)\, dw$$

$$= \int \text{sigm}(w^T x)\, N(w|m_N, V_N)\, dw$$

$$= \int \text{sigm}(a)\, N(a|\mu_a, \sigma_a^2)\, da$$

$$a \triangleq w^T x$$

$$\mu_a \triangleq E[a] = m_N^T x$$

$$\sigma_a^2 = var[a] = \int p(a|D)[a^2 - E[a^2]]\, da$$

$$= \int p(w|D)[(w^T x)^2 - (m_N^T x)^2]\, dw$$

$$= x^T V_N x$$

$$\Phi(a) \triangleq \int_{-\infty}^{a} N(x|0,1)\, x$$

$$\int \Phi(\lambda a) N(a|\mu, \sigma^2)\, da = \Phi\left(\frac{a}{(\lambda^{-2} + \sigma^2)^{\frac{1}{2}}}\right)$$

$$sigm(a) \approx \Phi(\lambda a)$$

$$\int sigm(a) N(a|\mu, \sigma^2)\, da \approx sigm(k(\sigma^2)\mu)$$

$$k(\sigma^2) \triangleq (1 + \pi\sigma^2/8)^{-\frac{1}{2}}$$

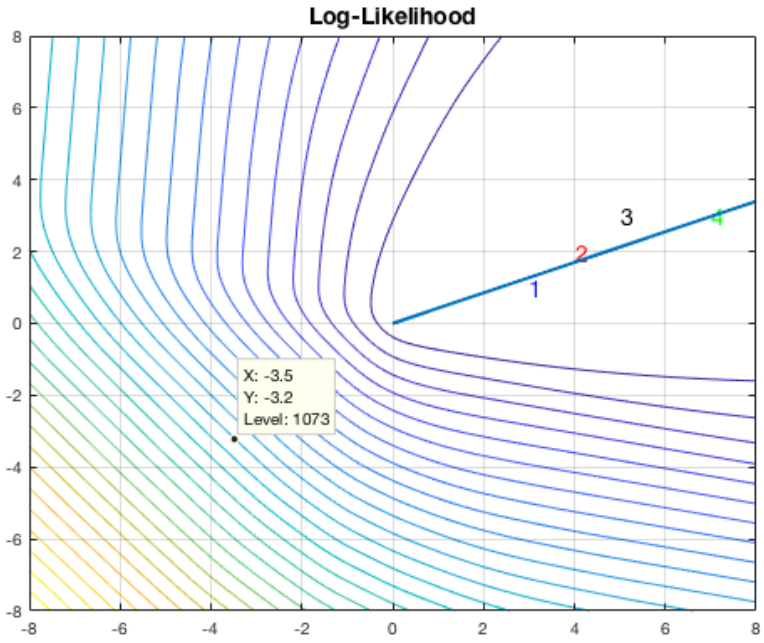
Therefor

$$p(y=1|x, D) \approx sigm(k(\sigma^2)\mu_a)$$

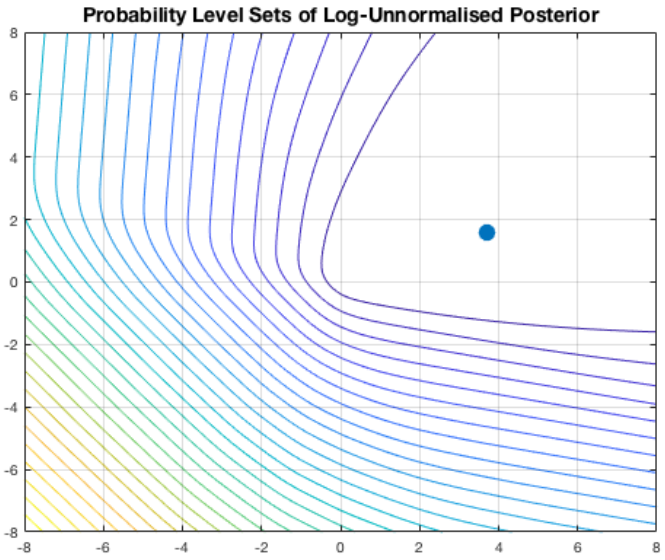
$$0 \leq k(\sigma^2) \leq 1$$

$$sigm(k(\sigma^2)\mu) \leq sigm(\mu) = p(y=1|x, w_{MAP})$$

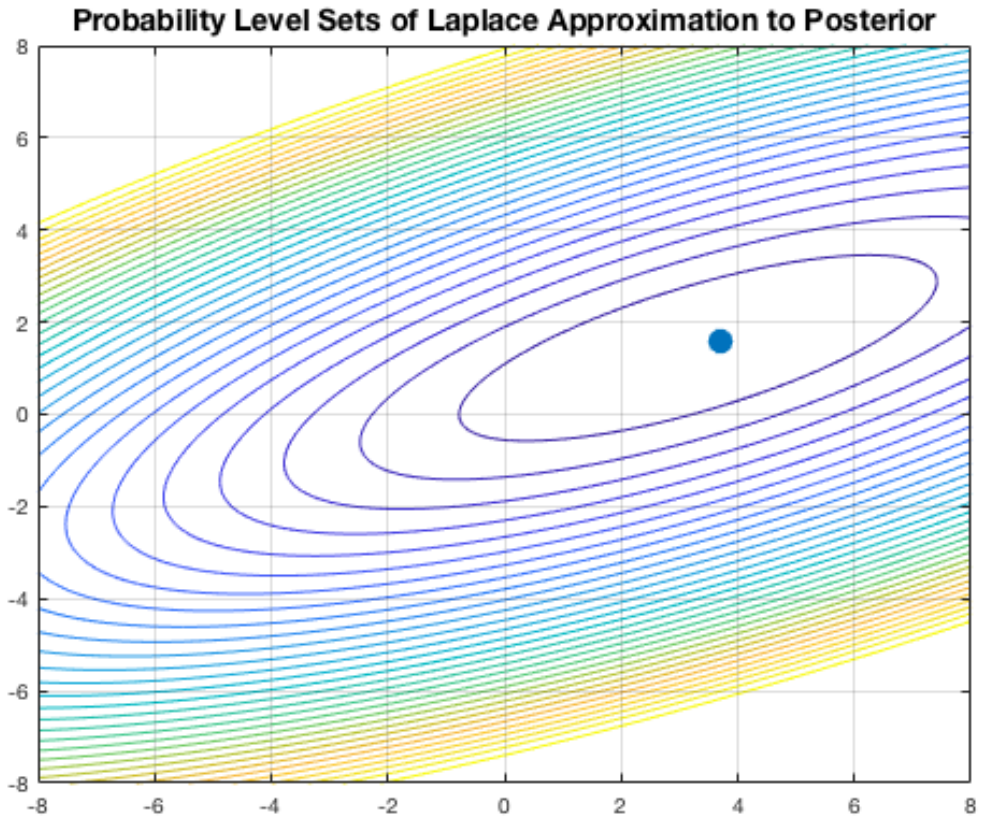
(c) log-likelihood




log-unnormalised posterior



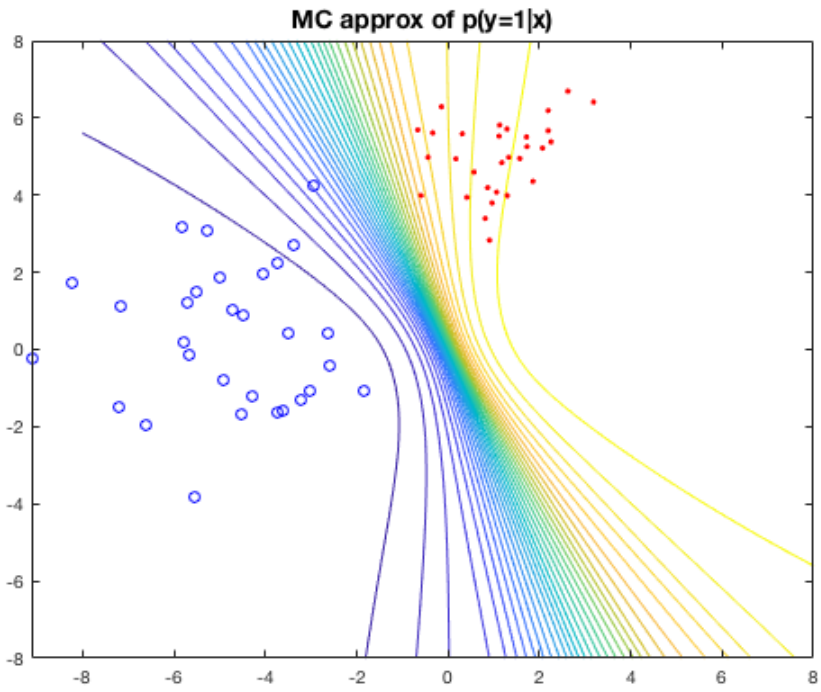
# Laplace Approximation




using the pmtk3 matlab code

(d)

MC approximation




Probit approximation